108

Some Easily Analyzable Convolutional Codes

R. McEliece, S. Dolinar, and F. Pollara
Communications Systems Research Section

H. Van Tilborg
Eindhoven University, Mathematics Department, The Netherlands

Convolutional codes have played and will play a key role in the downlink telemetry systems on many NASA deep-space probes, including Voyager, Magellan, and Galileo. One of the chief difficulties associated with the use of convolutional codes, however, is the notorious difficulty of analyzing them. Given a convolutional code as specified, say, by its generator polynomials, it is no easy matter to say how well that code will perform on a given noisy channel. The usual first step in such an analysis is to compute the code's free distance; this can be done with an algorithm whose complexity is exponential in the code's constraint length. The second step is often to calculate the transfer function in one, two, or three variables, or at least a few terms in its power series expansion. This step is quite hard, and for many codes of relatively short constraint length, it can be intractable. However, we have discovered a large class of convolutional codes for which the free distance can be computed by inspection, and for which there is a closed-form expression for the three-variable transfer function. Although for large constraint lengths, these codes have relatively low rates, they are nevertheless interesting and potentially useful. Furthermore, the ideas developed here to analyze these specialized codes may well extend to a much larger class.

I. Introduction

In this article a class of binary $(n, 1)$, constraint length K, convolutional codes, called *zero-run length (ZRL) convolutional codes*, is defined and studied. These codes are interesting because they are easy to analyze. ZRL codes include as special cases orthogonal convolutional codes, the recent "superorthogonal codes" of Viterbi, and many others. None of the convolutional codes currently used in NASA missions belong to the ZRL class. For any ZRL code, it is possible to compute the free distance by inspection, and to write down the complete transfer function

$T(D, I, L)$, explicitly (see Theorem 7, below). Important variations of the transfer function, viz.

$$T_{\text{num}}(D) = T(D, 1, 1)$$

$$T_{\text{bit}}(D) = \frac{\partial T}{\partial I}(D, 1, 1)$$

$$T_{\text{len}}(D) = \frac{\partial T}{\partial L}(D, 1, 1)$$

are commonly used to overbound the probability of decoder error for these codes ([3], Section 9.3, or [4], Section 4.4). For arbitrary convolutional codes, these functions can be very complicated indeed (see [7]), but for any ZRL code these functions have simple, closed-form expressions (see Corollary 8).

II. Zero-Run Length Convolutional Codes

Any $(n, 1)$, constraint length K convolutional code is characterized by a list of n generator polynomials $(g_1(x), \ldots, g_n(x))$, where $g_i(x) = g_{i,0} + g_{i,1}x + \cdots + g_{i,K-1}x^{K-1}$ is a polynomial of degree $K-1$ or less. The encoder for such a code consists of a shift register of length $K-1$, with one input and n outputs; the *state* of the encoder is defined to be the contents of the shift register. If $(s_1, \ldots, s_{K-1})$ is the current state, and s_0 is the current input, the next state is $(s_0, \ldots, s_{K-2})$ and the output, which we will call a *code segment*, is the n-tuple $(y_1, \ldots, y_n)$, where $y_i = \sum_{j=0}^{K-1} s_j g_{i,j}$.

1. Definition. An encoder state $s = (s_1 s_2 \cdots s_{K-1})$ is said to have *zero-run length* i, written "ZRL$(s) = i$" for short, if s contains exactly i leading zeros. For example, with $K = 5$, ZRL(0010) = 2, ZRL(0000) = 4, and ZRL(1001) = 0. In general, for an $(n, 1)$, constraint length K, convolutional code, there will be 2^{K-1} states, but only K possible values for ZRL $(0, 1, \ldots, K-1)$.

Note that if the encoder is in a state of zero-run length i, and the input is 0, the next state will have ZRL $= \min(i+1, K-1)$, whereas if the input is 1, the next state will have ZRL = 0. Thus the ZRL of the encoder's next state depends only on the current value of ZRL and the input. This fact is illustrated in Fig. 1, which shows the topology of states, organized according to the values of ZRL. In Fig. 1, the arrows marked with α's represent state transitions caused by 0 inputs, and the arrows marked with β's represent state transitions caused by 1 inputs. We will return to this state diagram in the proof of our main result, Theorem 7, below.

2. Definition. An $(n, 1)$ convolutional code of constraint length K is said to be a *ZRL* code if the output weight depends only on the input and the ZRL of the state. The symbol u_i is used to denote the output weight if the encoder has ZRL $= i$ and the input is 0, and the symbol w_i is used if ZRL $= i$ and the input is 1. The u_i's and the w_i's are conveniently displayed in a $2 \times K$ matrix, called the *weight matrix* of the code:

$$W = \begin{array}{c} \\ 0 \\ 1 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & \cdots & K-1 \end{array} \\ \begin{pmatrix} u_0 & u_1 & \cdots & u_{K-1} \\ w_0 & w_1 & \cdots & w_{K-1} \end{pmatrix} \end{array}$$

3. Example. Let $K = 3$. Then the $(4, 1)$ convolutional code with generator polynomial list $(1, x, 1 + x^2, 1 + x + x^2)$ is a ZRL code. Since with $K = 3$ there is only one state with ZRL = 1, viz. 01, and only one state with ZRL = 2, viz. 00, in order to verify that this code is ZRL, one need only investigate the two states with ZRL = 0, i.e., 10 and 11. If the state is 10 and the input is 0, the output is (0101), whereas if the input is 1 the output is (1110). On the other hand, if the state is 11 and the input is 0, the output is (0110), and if the input is 1, the output is (1101). Thus, if the state has ZRL = 0, and the input is 0, the output weight is 2; and if the input is 1, the output weight is 3. Hence, the output weight indeed depends only on the state's ZRL, as required. The weight table for this code is as follows:

$$W = \begin{array}{c} \\ 0 \\ 1 \end{array} \begin{array}{c} \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \begin{pmatrix} 2 & 2 & 0 \\ 3 & 1 & 3 \end{pmatrix} \end{array}$$

4. Definition. The *profile* of an $(n, 1)$, constraint length K ZRL convolutional code is the vector $(d_1, d_2, \ldots, d_K)$, where d_i is the Hamming weight of the output of the encoder, beginning in a state with ZRL = 0, with length i input sequence $0^{i-1}1$.

5. Lemma. In terms of the entries in the weight table, the profile of a ZRL convolutional code is

$$d_i = u_0 + u_1 + \cdots + u_{i-2} + w_{i-1}$$

for $i = 1, 2, \ldots, K$

Proof: If one starts in a state with ZRL = 0, and uses the input sequence 0^{i-1}, one passes through states

with ZRL $= 1, 2, \ldots, i-2$, causing outputs of weight $u_0, u_1, \ldots, u_{i-2}$, and arrives at a state with ZRL $= i-1$. The last input of 1 causes the encoder to move to a state with ZRL $= 0$ and to produce an output of weight w_{i-1}.

6. Example. Combining the weight table in Example 3 with Lemma 5, one finds that the profile of the code in Example 3 is $(3,3,7)$: $d_1 = w_0 = 3$; $d_2 = u_0 + w_1 = 2+1 = 3$; and $d_3 = u_0 + u_1 + w_2 = 2+2+3 = 7$.

III. Transfer Function for ZRL Codes

The following theorem is our main result. It gives the promised closed-form expression for the transfer function of a ZRL code in terms of its profile.

7. Theorem. For a ZRL convolutional code with profile $(d_1, \ldots, d_K)$, the three-variable transfer function is given by

$$T(D,I,L) = \frac{D^{d_K} I L^K}{1 - \sum_{i=1}^{K-1} D^{d_i} I L^i}$$

Proof: One begins by reviewing the definition of $T(D, I, L)$ for an arbitrary $(n,1)$, constraint length K, convolutional code. (See [3] or [4] for more details.)

Starting with the state diagram for the given code, which is the 2^{K-1} vertex deBruijn graph, each of the 2^K edges is labelled with a monomial in the three indeterminates D, I, and L, i.e., a term of the form $D^w I^\epsilon L$. The power w of D in the monomial represents the Hamming weight of the encoder output corresponding to the given state transition, and ϵ is either 0 or 1, according to whether the corresponding encoder input is zero or one. The resulting labelled, directed graph is called the "DIL state diagram" for the code.

In Fig. 2 is the DIL state diagram for a $K = 3$ ZRL code. For example, in Fig. 2 the edge from state 10 to 11 is labelled $D^{w_0} IL$. This is because the transition $10 \rightarrow 11$ is caused by an encoder input of 1, so that the exponent of I in the edge label is 1. State 10 has ZRL $= 0$, and by definition of a ZRL code, when the state has ZRL $= 0$ and the input is 1, the output weight is w_0; thus the exponent on D in the label is w_0. The other seven edge labels can be explained similarly.

A *path* of length m in the DIL state diagram is defined as a sequence of $m+1$ vertices such that each adjacent pair of vertices in the sequence is connected by a directed edge. For example, in Fig. 2, the vertex sequence $00 \rightarrow 10 \rightarrow 01 \rightarrow 00$ is a path of length 3. A path is completely specified by its initial vertex and the string of input bits corresponding to the vertex transitions, which we call the *input string* of the path. For example, the path $00 \rightarrow 10 \rightarrow 01 \rightarrow 00$ has initial vertex 00 and input string 100. The *weight* of a path is defined to be the product of the labels on its edges. For example, the path $00 \rightarrow 10 \rightarrow 01 \rightarrow 00$ in Fig. 2 has weight $D^{w_2+u_0+u_1} IL^3$.

The three-variable transfer function $T(D, I, L)$ is now defined to be *the sum of the weights of all paths from vertex* 0^{K-1} *back to vertex* 0^{K-1} *which have no intermediate returns to vertex* 0^{K-1}. Alternatively, $T(D, I, L)$ is the sum of the weights of all paths with initial vertex 0^{K-1} whose input string ends with 0^{K-1} but has no other substring equal to 0^{K-1}. (In [3, Section 9.3] these paths are called "fundamental paths.")

In principle, one can compute $T(D, I, L)$ for any convolutional code using the standard "transfer matrix method" described, for example, in [5, Sec. 4.7]. However, this method is essentially equivalent to inverting a $2^{K-1} \times 2^{K-1}$ matrix with three-variable monomial entries, and is not in general practical except for codes with extremely small constraint lengths [7]. However, for a ZRL code, one can simplify this calculation considerably, by first "collapsing" the state diagram by combining states with the same value of ZRL. In the collapsed state diagram, there will be K vertices, labelled $0, 1, \ldots, K-1$; vertex i will be connected by a directed edge to vertex j if there is any edge in the original (noncollapsed) DIL state diagram connecting a vertex with ZRL $= i$ to one with ZRL $= j$. The label on an edge in the reduced state diagram will be the same as the label on the corresponding edge in the original graph; the ZRL property implies that this rule is well defined.

The collapsing process is illustrated in Fig. 3, which shows the collapsed version of the graph in Fig. 2. Note, for example, that in Fig. 3 the edge from vertex 0 to vertex 1 is labelled $D^{u_0} L$. This is because in Fig. 2, both edges from a vertex with ZRL $= 0$ to a vertex with ZRL $= 1$, viz. $10 \rightarrow 01$ and $11 \rightarrow 01$, have the same label $D^{u_0} L$.

When the DIL state diagram for a constraint length K ZRL code is collapsed, the resulting state diagram will be identical to the state diagram in Fig. 1, where the labels α_i and β_i are given by

$$\alpha_i = D^{u_i} L$$

$$\beta_i = D^{w_i} IL$$

One can think of the collapsed state diagram of Fig. 1 as the state diagram of a finite-state machine, with input alphabet $\{0,1\}$ and output alphabet the set of monomials $D^w I^c L$. If this machine is in state i and its input is 0, its next state is $\min(i+1, K+1)$, and its output is $D^{u_i} L$; if it is in state i and its input is 1, its next state is 0 and its output is $D^{w_i} IL$. Note that, as for the original state diagram, any path in the collapsed state diagram is specified by its initial vertex and its input string. For example, the path $2 \to 0 \to 1 \to 2$ in the collapsed state diagram of Fig. 3 has initial vertex 2 and input string 100. Its weight is $D^{w_2+u_0+u_1} L^3 I$.

The important point is that the collapsed state diagram is equivalent to the original state diagram for purposes of computing the $T(D,I,L)$ transfer function for the ZRL code. This is because a path in the original DIL state diagram with an initial vertex with ZRL $= i$ and input string σ will have the same weight as a path in the collapsed state diagram with initial vertex i and the same input string σ. For example, the path in the state diagram of Fig. 2 with initial vertex 00 and input string 100 has weight $D^{w_2+u_0+u_1} L^3 I$, which is the same as the weight of the path in the collapsed state diagram of Fig. 3 with initial state 2 and input string 100.

It follows then that the $T(D,I,L)$ transfer function for a ZRL code is the sum of the weights of all paths in the collapsed state diagram of Fig. 1 from state $K-1$ back to state $K-1$, with no intermediate returns to state $K-1$. This transfer function is denoted by $T^*_{K-1,K-1}$. One way to compute $T^*_{K-1,K-1}$ is to remove the vertices $1, 2, \ldots, K-2$ from the state diagram, but to preserve the path label information by relabelling the remaining edges appropriately, as shown in Fig. 4. For example, in Fig. 4, the edge from vertex 0 to vertex $K-1$ is labelled $\alpha_0 \alpha_1 \cdots \alpha_{K-2}$; this is because in Fig. 1 there is exactly one path from vertex 0 to vertex $K-1$ that uses only the deleted vertices $\{1, 2, \ldots, K-1\}$, viz. $012\cdots(K-1)$, and its weight is $\alpha_0 \alpha_1 \cdots \alpha_{K-2}$. Similarly, the loop at vertex 0 is relabelled to reflect the fact that there are $K-1$ paths from vertex 0 back to vertex 0 which use only the deleted vertices: $00, 010, 0120, \ldots, 012\cdots(K-2)0$, and the sum of the weights of these $K-1$ paths is $\beta_0 + \alpha_0\beta_1 + \cdots + \alpha_0 \cdots \alpha_{K-3}\beta_{K-2}$, which is the label on the loop at vertex 0 in Fig. 4.

Once the state diagram has been reduced to only two states, the computation of the transfer function $T^*_{K-1,K-1}$ is straightforward. Any path from vertex $K-1$ back to vertex $K-1$ with no intermediate return to vertex $K-1$ in Fig. 4 must be of the form $(K-1)0\cdots0(K-1)$, and so the desired transfer function is equal to the weight of the path $(K-1)0(K-1)$ divided by 1 minus the weight of the loop at vertex 0, i.e.,

$$T^*_{K-1,K-1} = \frac{\alpha_0\alpha_1\cdots\alpha_{K-2}\beta_{K-1}}{1-\beta_0-\alpha_0\beta_1-\alpha_0\alpha_1\beta_2-\cdots-\alpha_0\cdots\alpha_{K-3}\beta_{K-2}}$$

If one substitutes the above values for α_i and β_i into this expression, and uses the definition of the profile, the expression for $T(D,I,L)$ in the statement of the theorem is obtained.

8. Corollary. For a ZRL convolutional code with profile $(d_1, \ldots, d_K)$, the free distance is d_K and

$$T_{\text{num}}(D) = \frac{D^{d_K}}{P(D)}$$

$$T_{\text{bit}}(D) = \frac{D^{d_K}}{P(D)^2}$$

$$T_{\text{len}}(D) = \frac{D^{d_K} Q(D)}{P(D)^2}$$

where the polynomials $P(D)$ and $Q(D)$ are defined by

$$P(D) = 1 - \sum_{i=1}^{K-1} D^{d_i}$$

$$Q(D) = K - \sum_{i=1}^{K-1} (K-i) D^{d_i}$$

Proof: This follows directly from Theorem 7 and the definitions of $T_{\text{num}}(D)$, $T_{\text{bit}}(D)$, and $T_{\text{len}}(D)$ given at the beginning of the article.

9. Example. Continuing Examples 3 and 6, the profile is $(3,3,7)$, and so $P(D) = 1-2D^3$, $Q(D) = 3-3D^3$. Thus, by Corollary 8, $d_{\text{free}} = 7$, and

$$\begin{aligned} T_{\text{num}}(D) &= \frac{D^7}{1-2D^3} \\ &= D^7 + 2D^{10} + 4D^{13} \\ &\quad + 8D^{16} + 16D^{19} + 32D^{22} + \cdots \end{aligned}$$

$$T_{\text{bit}}(D) = \frac{D^7}{(1-2D^3)^2}$$
$$= D^7 + 4D^{10} + 12D^{13} + 32D^{16} + 80D^{19} + 192D^{22} + \cdots$$

$$T_{\text{len}}(D) = \frac{D^7(3-3D^3)}{(1-2D^3)^2}$$
$$= 3D^7 + 9D^{10} + 24D^{13} + 60D^{16} + 144D^{19} + 336D^{22} + \cdots$$

IV. Superorthogonal and Ultraorthogonal Codes

Next, two important general classes of ZRL convolutional codes, the superorthogonal codes introduced by Viterbi [1] and the ultraorthogonal codes introduced here, are defined.

10. Definition. The *superorthogonal code* of constraint length K, denoted by S_K, is defined as follows: $S_1 = (1)$, and for $K \geq 2$, then S_K is a $(2^{K-2}, 1)$ code whose generator polynomials are all 2^{K-2} possible polynomials of the form $1 + g_1 x + \cdots + g_{K-2}x^{K-2} + x^{K-1}$.

11. Definition. The *ultraorthogonal code* of constraint length K, denoted by U_K, is defined as follows: $U_1 = (0)$, and for $K \geq 2$, then U_K is a $(2^{K-2}, 1)$ code whose generator polynomials are all 2^{K-2} possible polynomials of the form $g_1 x + \cdots + g_{K-2}x^{K-2} + x^{K-1}$.

12. Example. For $K = 3$ the code S_3 has generator polynomial list $(1 + x^2, 1 + x + x^2)$, and U_3 has generator polynomial list $(x^2, x + x^2)$.

13. Theorem. For all $K \geq 1$, the codes S_K and U_K are ZRL codes. The weight tables for the superorthogonal codes are as follows:

$$W(S_1) = \begin{array}{c} \\ 0 \\ 1 \end{array} \begin{array}{c} 0 \\ \begin{pmatrix} 0 \\ 1 \end{pmatrix} \end{array}$$

$$W(S_2) = \begin{array}{c} \\ 0 \\ 1 \end{array} \begin{array}{c} 0 \quad 1 \\ \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \end{array}$$

$$W(S_3) = \begin{array}{c} \\ 0 \\ 1 \end{array} \begin{array}{c} 0 \quad 1 \quad 2 \\ \begin{pmatrix} 1 & 2 & 0 \\ 1 & 0 & 2 \end{pmatrix} \end{array}$$

and, for $K \geq 3$

$$W(S_K) = \begin{array}{c} \\ 0 \\ 1 \end{array} \begin{array}{c} 0 \quad 1 \quad \cdots \quad K-3 \quad K-2 \quad K-1 \\ \begin{pmatrix} 2^{K-3} & 2^{K-3} & \cdots & 2^{K-3} & 2^{K-2} & 0 \\ 2^{K-3} & 2^{K-3} & \cdots & 2^{K-3} & 0 & 2^{K-2} \end{pmatrix} \end{array}$$

Similarly, the weight tables for the ultraorthogonal codes are as follows:

$$W(U_1) = \begin{array}{c} \\ 0 \\ 1 \end{array} \begin{array}{c} 0 \\ \begin{pmatrix} 0 \\ 0 \end{pmatrix} \end{array}$$

$$W(U_2) = \begin{array}{c} \\ 0 \\ 1 \end{array} \begin{array}{c} 0 \quad 1 \\ \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} \end{array}$$

$$W(U_3) = \begin{array}{c} \\ 0 \\ 1 \end{array} \begin{array}{c} 0 \quad 1 \quad 2 \\ \begin{pmatrix} 1 & 2 & 0 \\ 1 & 2 & 0 \end{pmatrix} \end{array}$$

and, for $K \geq 3$

$$W(U_K) = \begin{array}{c} \\ 0 \\ 1 \end{array} \begin{array}{c} 0 \quad 1 \quad \cdots \quad K-3 \quad K-2 \quad K-1 \\ \begin{pmatrix} 2^{K-3} & 2^{K-3} & \cdots & 2^{K-3} & 2^{K-2} & 0 \\ 2^{K-3} & 2^{K-3} & \cdots & 2^{K-3} & 2^{K-2} & 0 \end{pmatrix} \end{array}$$

Proof: The key to the proof is the close relationship between the convolutional codes S_K and U_K and the first-order Reed-Muller (1RM) block codes, which are now described. The $(2^m, m+1)$ 1RM code can be defined by an $(m+1) \times 2^m$ generator matrix G_m which has as columns all possible binary $(m+1)$-tuples ending with 1. For example, with $m = 2$ the $(4, 3)$ 1RM code has generator matrix

$$G_2 = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

It is known that all weights in the $(2^m, m+1)$ 1RM code are equal to 2^{m-1}, except for the all-zero word and the all-one word ([8], Chapter 13). If G_m^0 is defined to be the matrix obtained by adding a row of zeros at the top of G_m, and G_m^1 to be the matrix obtained by adding a row of ones at the top of G_m, then the columns of G_{K-2}^0 give the coefficients of the generator polynomials of U_K

and the columns of G^1_{K-2} give the generator polynomials of S_K. For example, again with $m = 2$,

$$G_2^0 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \qquad G_2^1 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

It therefore follows that every (2^{K-2})-bit code segment in either of the codes S_K or U_K is a word in the $(2^{K-2}, K-1)$ 1RM code. In almost every case, this segment will have weight 2^{K-3}; the only other possibilities are weight 0 (the all-zero codeword) and weight 2^{K-2} (the all-one codeword). To analyze these exceptional cases, note that every linear combination of rows of either G^0_m or G^1_m is a word in the 1RM code. All such linear combinations will therefore have weight 2^{m-1}, with the following exceptions. In G^0_m, the empty linear combination, or the top row, give the all-zero codeword; and the bottom row, or the top row plus the bottom row, give all ones. In G^1_m, the empty linear combination or the top row plus the bottom row gives the all-zero codeword; and the top row or the bottom row gives all ones.

Therefore, in the ultraorthogonal code U_K, the code segment will be all zeros if and only if the state is 0^{K-1} and the input is 0, or the state is 0^{K-1} and the input is 1. Similarly, the code segment will be all ones if and only if the state is $0^{K-2}1$ and the input is zero, or the state is $0^{K-2}1$ and the input is 1. Thus, the output weight will be 2^{K-2} unless the state has ZRL $= K-1$ and the input is 0 or 1, in which case the output weight is 0, or if the state has ZRL $= K-2$ and the input is 0 or 1, in which case the output weight is 2^{K-1}. This is what the theorem states about the ultraorthogonal codes.

Similarly, in the superorthogonal code S_K, the code segment will be all zeros if and only if the state is 0^{K-1} and the input is 0, or the state is $0^{K-2}1$ and the input is 1. Similarly, the code segment will be all ones if and only if the state is 0^{K-1} and the input is 1, or the state is $0^{K-2}1$ and the input is 0. Thus, the output weight will be 2^{K-2} unless the state has ZRL $= K-1$ and the input is 0, or if the state has ZRL $= K-2$ and the input is 1, in which case the output weight is 0; or if the state has ZRL $= K-1$ and the input is 1, or if the state has ZRL $= K-2$, and the input is 0, in which case the output weight is 2^{K-1}. This is what the theorem states about the superorthogonal codes.

Theorem 13 provides many ZRL codes. The following definition and the discussion that follows will show how to use the superorthogonal and ultraorthogonal codes to build many other ZRL codes.

14. Definition. Given two convolutional codes, their *sum* is defined to be the convolutional code whose generator polynomial (g.p.) list is obtained by merging the g.p. lists for the original codes. Thus for example, the sum of the $(3,1)$ code with g.p. list $(1, 1+x, 1+x+x^2)$ and the $(2,1)$ code with g.p. list $(1+x^2, 1+x+x^2)$ is the $(5,1)$ code with g.p. list $(1, 1+x, 1+x^2, 1+x+x^2, 1+x+x^2)$. In general, the sum of an $(n_1, 1)$ convolutional code of constraint length K_1 and an $(n_2, 1)$ convolutional code of constraint length K_2 is an $(n_1+n_2, 1)$ convolutional code of constraint length $\max(K_1, K_2)$.

15. Lemma. If C_1 and C_2 are ZRL convolutional codes, with constraint lengths K_1 and K_2, respectively, with $K_1 \leq K_2$, then $C_1 + C_2$ is also ZRL, and the weight table for $C_1 + C_2$ is obtained from the weight tables W_1 and W_2 by first extending W_1 by repeating its last column $K_2 - K_1$ times, and then adding the two weight tables together.

Proof: If the two codes have the same constraint length, this is immediate. If, however, the two constraint lengths are different, and $K_1 < K_2$, C_1 can nevertheless be regarded as a convolutional code with constraint length K_2 in which the last $K_2 - K_1$ bits in the shift register are never used. States with ZRL values $K_1, K_1+1, \ldots, K_2-1$, will plainly behave just like the all-zeros state (with ZRL $= K_1 - 1$), and the extra $K_2 - K_1$ columns that appear in the weight matrix will be identical to the last column of the unextended weight matrix. The result now follows.

16. Example. The code of Example 3 is $S_1+U_2+S_3$, as may easily be verified. The corresponding weight tables are, by Theorem 13,

$$W(S_1) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$W(U_2) = \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}$$

$$W(S_3) = \begin{pmatrix} 1 & 2 & 0 \\ 1 & 0 & 2 \end{pmatrix}$$

To obtain the weight matrix for $S_1 + U_2 + S_3$, first extend $W(S_1)$ and $W(U_2)$ to dimensions 2×3 by repeating the

respective last rows, and then adding the resulting matrices:

$$W = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix} + \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 1 & 2 & 0 \\ 1 & 0 & 2 \end{pmatrix} = \begin{pmatrix} 2 & 2 & 0 \\ 3 & 1 & 3 \end{pmatrix}$$

which is the same as was seen in Example 3.

17. Example. For any K, the code $\sum_{i=1}^{K}(S_i + U_i)$ is by Lemma 15 a ZRL code. In fact, this code has as generator polynomials all 2^K polynomials of degree $\leq K-1$; it is the orthogonal code of constraint length K.

18. Theorem. The profiles of the codes S_K are:

$$\begin{aligned}
&\text{profile}(S_1) = (1) \\
&\text{profile}(S_2) = (0, 2) \\
&\text{profile}(S_3) = (1, 1, 5) \\
&\text{profile}(S_4) = (2, 4, 4, 12) \\
&\text{profile}(S_5) = (4, 8, 12, 12, 28) \\
&\qquad\vdots \\
&\text{profile}(S_K) = (2^{K-3}, 2 \cdot 2^{K-3}, \ldots, \\
&\qquad (K-2)2^{K-3}, (K-2)2^{K-3}, \\
&\qquad (K+2)2^{K-3})
\end{aligned}$$

The profiles of the codes U_K are

$$\begin{aligned}
&\text{profile}(U_1) = (0) \\
&\text{profile}(U_2) = (1, 1) \\
&\text{profile}(U_3) = (1, 3, 3) \\
&\text{profile}(U_4) = (2, 4, 8, 8) \\
&\text{profile}(U_5) = (4, 8, 12, 20, 20) \\
&\qquad\vdots \\
&\text{profile}(U_K) = (2^{K-3}, 2 \cdot 2^{K-3}, \ldots, \\
&\qquad (K-2)2^{K-3}, K2^{K-3}, K2^{K-3})
\end{aligned}$$

Proof: This follows by combining Theorem 13 and Lemma 5.

19. Example. By combining Theorems 7 and 18, one can obtain the transfer function for the superorthogonal codes. Indeed, if $z = D^{2^{K-3}}$, it follows from these theorems that for the superorthogonal code of constraint length K,

$$\begin{aligned}
&T(D, I, L) \\
&= \frac{z^{K+2} I L^K}{1 - zIL(1 + zL + \cdots + z^{K-3}L^{K-3}) - z^{K-2}IL^{K-1}} \\
&= \frac{z^{K+2} I L^K (1 - zL)}{1 - z(L + IL) - z^{K-2}IL^{K-1} + z^{K-1}(IL^{K-1} + IL^K)}
\end{aligned}$$

an expression first found by Viterbi [1]. It follows then from Corollary 8 that $d_{\text{free}} = (K+2)2^{K-3}$ and

$$\begin{aligned}
T_{\text{num}}(D) &= \frac{z^{K+2}(1-z)}{1 - 2z - z^{K-2} + 2z^{K-1}} \\
&= \frac{z^{K+2}(1-z)}{(1-2z)(1-z^{K-2})} \\
&= z^{K+2}\left\{ \frac{2^{K-3}}{(2^{K-2}-1)(1-2z)} \right. \\
&\qquad \left. + \frac{(2^{K-3}-1) - z - 2z^2 - \cdots - 2^{K-4}z^{K-3}}{(2^{K-2}-1)(1-z^{K-2})} \right\}
\end{aligned}$$

In the last expression, a two-term partial-fraction decomposition is seen (in braces) for the generating function $T_{\text{num}}(D)/z^{K+2}$. The coefficient of z^k in the expansion of the first term is

$$\frac{2^{K-3}}{2^{K-2}-1} \cdot 2^k$$

The coefficients of the expansion of the second term are periodic of period $K-2$, and each term is less than $1/2$ in absolute value. Since it is known that the coefficient of z^k in the combined expansion is an integer, it follows that this coefficient must be the integer closest to

$$\frac{2^{K-3}}{2^{K-2}-1} \cdot 2^k$$

Therefore, it has been proved that the coefficient of $D^{d_{\text{free}}+k2^{K-3}}$ in $T_{\text{num}}(D)$ for the superorthogonal code of constraint length K is

$$N_{d_{\text{free}}+k2^{K-3}} = \text{integer closest to } \frac{2^{K-3}}{2^{K-2}-1} \cdot 2^k$$

As a special case, it is found that the $(8,1)$, $K = 5$ superorthogonal code has $d_{\text{free}} = 28$, and the number of fundamental paths of weight $28 + 4k$ is the integer closest to $\frac{4}{7} \cdot 2^k$, i.e.,

$$T_{\text{num}}(D) = D^{28} + D^{32} + 2D^{32} + 5D^{36} + 9D^{40} + 18D^{44} + 37D^{48} + O(D^{52})$$

V. A Representation Theorem

If Theorem 13 is combined with Lemma 15, many ZRL codes can be constructed. It is surprising (and perhaps disappointing) that all such codes are constructed this way.

20. Theorem. An $(n,1)$ convolutional code C of constraint length K is ZRL if and only if it is the sum of copies of superorthogonal and ultraorthogonal codes:

$$C = \sum_{i=1}^{K} (m_i S_i + n_i U_i)$$

where m_i and n_i are integers denoting the multiplicities of S_i and U_i in the code C.

Proof: The proof of this theorem is lengthy and will be omitted.

The next lemma, when combined with Theorems 20 and 18, enables one to write down the transfer functions for any ZRL convolutional code.

21. Lemma. If C_1 and C_2 are ZRL convolutional codes, with constraint lengths K_1 and K_2 respectively, with $K_1 \leq K_2$, then the profile for the sum $C_1 + C_2$ is obtained from the profiles for C_1 and C_2 by first extending profile(C_1) to length K_2 by repeating its last entry $K_2 - K_1$ times, and then adding the two profiles together.

Proof: This follows by combining Lemma 15 with Lemma 5.

22. Example. The ZRL code in Example 3 is $C = S_1 + U_2 + S_3$, as was seen in Example 16. The corresponding profiles are, by Theorem 19,

$$\text{profile}(S_1) = (1)$$

$$\text{profile}(U_2) = (1,1)$$

$$\text{profile}(S_3) = (1,1,5)$$

To obtain C's profile, use Lemma 21. First extend the profiles of S_1 and U_2 to length 3 by repeating the last entries, and then add the resulting lists:

$$\text{profile}(C) = (1,1,1) + (1,1,1) + (1,1,5) = (3,3,7)$$

as was seen in Example 6. However, for the same values of n and K, one can get a larger d_{free} by considering the code $2S_3$ instead, since its profile is $2(1,1,5) = (2,2,10)$, so that $d_{\text{free}} = 10$. And in fact, for $n = 4$ and $K = 3$ this is the largest possible free distance, since the Plotkin bound for these parameters gives $d_{\text{free}} \leq 10$. In general, for $(n,1)$, $K = 3$ ZRL codes, the largest possible d_{free} is $\lfloor \frac{5n}{2} \rfloor$, achieved by $\lfloor \frac{n}{2} \rfloor S_3 + (n \bmod 2) S_2$, whereas the best possible d_{free} among all codes, ZRL or not, is $\lfloor \frac{8n}{3} \rfloor$, achieved by $\lfloor \frac{n+1}{3} \rfloor (1 + x^2) + \lfloor \frac{2n+1}{3} \rfloor (1 + x + x^2)$. The ratio of these two values approaches 16/15 as $n \to \infty$, and the smallest value of n for which these two values differ by as much as two is $n = 9$, where the best ZRL code $4S_3 + S_2$ has $d_{\text{free}} = 22$, but the code with g.p. list $(3(1 + x^2), 6(1 + x + x^2))$ has $d_{\text{free}} = 24$. However, even in this case the ZRL code may be competitive, since its T_{num} is

$$\frac{D^{22}}{1 - D^4 - D^6} = D^{22} + D^{26} + D^{28} + D^{30} + O(D^{32})$$

whereas the unrestricted code has

$$T_{\text{num}} = \frac{D^{24}(2 - D^6)}{1 - 3D^6 + D^{12}} = 2D^{24} + 5D^{30} + O(D^{36})$$

And indeed, an asymptotic analysis shows the rate of growth of the coefficients of $T_{\text{num}}(D)$ for the ZRL code to be $\approx (1.1577)^n$, whereas for the unrestricted code it is $\approx (1.1740)^n$. Thus, as discussed in [2], the ZRL code may perform better at low signal-to-noise ratios than the non-ZRL code.

VI. Summary

A class of convolutional codes, termed zero-run length (ZRL) convolutional codes, has been discovered for which the free distance can be computed by inspection, and for which there is a closed-form expression for the three-variable transfer function. This class of codes includes the superorthogonal codes introduced by Viterbi [1] and analogous "ultraorthogonal" codes introduced here. It has been found that, while ZRL codes are much more general than superorthogonal or ultraorthogonal codes, any ZRL code may be constructed as a combination ("sum") of superorthogonal and ultraorthogonal codes.

Although ZRL codes have very low rates for large constraint lengths, they are nevertheless interesting and potentially useful. Furthermore, many of the ideas developed here to analyze this class of specialized codes, such as the use of reduced state diagrams, might extend to other interesting code classes as well.

References

[1] A. J. Viterbi, "A New Class of Very Low Rate Convolutional Codes with Application to Spread Spectrum Multiple Access," preprint (April 1989).

[2] C.-C. Chao and R. J. McEliece, "On the Path Weight Enumerators of Convolutional Codes," Proc. 26th Ann. Allerton Conference, Univ. Illinois, pp. 1049–1058, October 1988.

[3] R. J. McEliece, *The Theory of Information and Coding*, Reading, Massachussetts: Addison Wesley, 1977.

[4] A. J. Viterbi and J. K. Omura, *Principles of Digital Communication and Coding*, New York: McGraw-Hill, 1979.

[5] R. P. Stanley, *Enumerative Combinatorics, Vol. I*, Monterey, California: Wadsworth & Brooks Cole, 1986.

[6] R. J. McEliece, R. B. Ash, and C. Ash, *Introduction to Discrete Mathematics*, Boston: Random House, 1989.

[7] I. Onyszchuk, "Efficient methods for computing transfer functions for convolutional codes," in preparation.

[8] F. J. MacWilliams and and N. J. A. Sloane, *The Theory of Error-Correcting Codes*, Amsterdam: North-Holland, 1977.

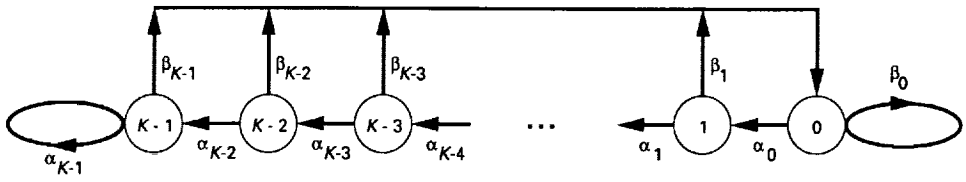

Fig. 1. Reduced state diagram for analyzing ZRL codes.

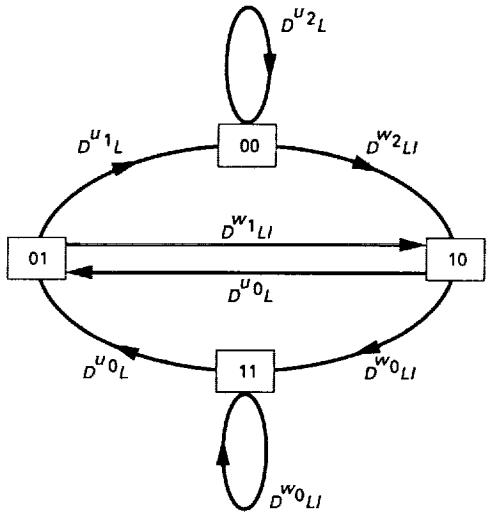

Fig. 2. The DIL state diagram for a K = 3 ZRL code.

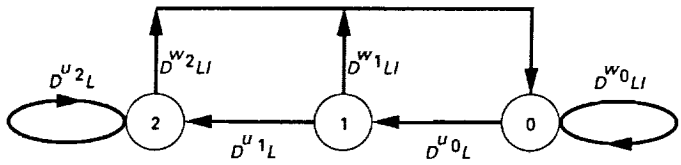

Fig. 3. The collapsed DIL state diagram for a K = 3 ZRL code (compare to Fig. 2).

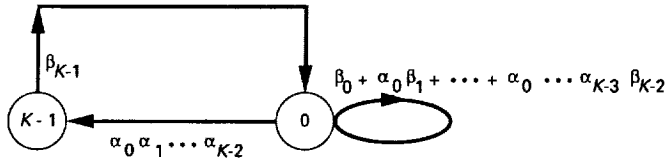

Fig. 4. The state diagram of Fig. 1, after the loop at state K−1 and the states 1, 2, . . ., K−2 have been eliminated.